Tuning The Trimming Factor

For arbitrary lower and upper bounds lb and ub, there are certain values of the trimming factor that can negatively impact correctness and query efficiency.[1] In this document, we describe two mappings that, given any domain, will output an appropriate trimming factor that maintains correctness, while achieving a good trade-off between query latency, insertion throughput, and storage overhead. In the following, we assume that there exist a default trimming factor $\mathsf{tf} \in \mathbb{N}_{\geq 0}$ as well as a default sparsity factor $\mathsf{sp} \in \{1, 2, 4, 8\}$.[2]

The integer case. The mapping , map_I, applies to the following numerical data types: `sint32`, `int64` and `sint64` and is defined as follows.

$$\mathsf{map}_I(\mathsf{lb}, \mathsf{ub}) = \begin{cases} \perp & \text{if } \mathsf{ub} < \mathsf{lb} \\ \min\left(\mathsf{tf}, \left\lfloor \mathsf{sp}^{-1} \cdot \left(\left\lceil \log(\mathsf{ub} - \mathsf{lb} + 1)\right\rceil\right)\right\rfloor\right) & \text{otherwise.} \end{cases}$$

where lb and ub belong to the underlying domain of the numerical data type.

The floating-point format case. The mapping , map_F, applies to the following numerical data types: `bin128` and `dec128` and is defined as follows.

$$\mathsf{map}_F(\mathsf{lb}, \mathsf{ub}, \mathsf{prc}) = \begin{cases} \perp & \text{if } \mathsf{ub} < \mathsf{lb} \\ \min\left(\mathsf{tf}, \left\lfloor \mathsf{sp}^{-1} \cdot \left(\left\lceil \log((\mathsf{ub} - \mathsf{lb} + 1) \cdot 10^{\mathsf{prc}})\right\rceil\right)\right\rfloor\right) & \text{otherwise.} \end{cases}$$

where lb and ub belong to the underlying domain of the numerical data type and where $\mathsf{prc} \in \mathbb{N}_{\geq 0}$.

[1] For example, if both the domain and the chosen trimming factor are large, then at query time, there are cases where the size of the cover can be extremely large. A very large cover has two implications: an efficiency implication as the number of binary hops is proportional to the size of the cover; and a correctness implication, as the number of generated tags in this case can be too high exceeding the `BSON` limit.

[2] Our experiments show that setting the trimming factor to 6 and the sparsity factor to 2 achieves the best trade-off between query latency and insertion throughput.