Quantile Sketch Algorithms

Levente Birszki

Supervisors:

Gábor rétvári
Balázs Vass

Eötvös Loránd University

2024 May 30

Motivation

- **Financial Analysis**: Quickly estimating value at risk (VaR) and other financial metrics from large volumes of transaction data.
- **Network Monitoring**: Analyzing latency, bandwidth usage, and other network metrics in real-time.
- **Database Systems**: Enhancing query performance by maintaining approximate summaries of large tables.

Basic concepts

Definition (sketch)

A *sketch* $S(X)$ of some data set X with respect to some function f is a compression of X that allows us to compute, or approximately compute $f(X)$ given access only to $S(X)$.

Definition (rank)

Given an x element from the input stream. $r(x)$, the *rank* of x is the number of elements smaller or equal than x in the sorted input.

Definition (quantile)

The q-quantile for $q \in [0, 1]$ is the element x_q, whose rank is $\lceil qn \rceil$.

Why sketches?

- **Scalability**: Traditional methods for computing quantiles can be impractical for large datasets due to high computational and storage costs.
- **Stream Processing**: In many real-time applications, data arrives in streams, and it's crucial to compute quantiles without storing the entire dataset.

Definition (rank error)

An element $\tilde{x}_q$ is an ϵ-approximate q-quantile if $|r(x_q) - r(\tilde{x}_q)| \leq \epsilon n$. This also known as rank error.

Former results

Definition (single quantile approximation problem)

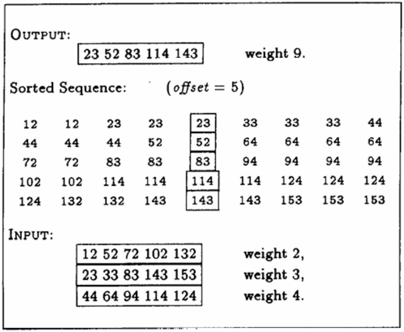
In the *single quantile approximation problem*, given an $x_1, \ldots, x_n$ input stream, q, ϵ and δ. Construct a streaming algorithm, which computes an ϵ-approximate q-quantile with probability at least $1 - \delta$.

Publication	Algorithm	Space Complexity	Mergeability	quantile type
2001	GK-sketch	$O\left(\frac{1}{\epsilon}\log(\epsilon n)\right)$	no	all
2004	q-digest	$O(\frac{1}{\epsilon}\log u)$	yes	all
2016	KLL	$O(\frac{1}{\epsilon}\log^2\log\frac{1}{\delta})$	yes	singe
2016	KLL	$O(\frac{1}{\epsilon}\log^2\log\frac{1}{\delta\epsilon}))$	yes	all
2017	FO	$O(\frac{1}{\epsilon}\log\frac{1}{\epsilon})$	no	all
2019	SweepKLL	$O(\frac{1}{\epsilon}\log\log\frac{1}{\delta}))$	no	single
2019	SweepKLL	$O(\frac{1}{\epsilon}\log\log\frac{1}{\delta\epsilon}))$	no	all

MRL-sketch framework

b buffers, each can store k elements. each buffer X has a $w(X)$ weight. Three operations:

- *New*(X): Fills an empty buffer from input, $w(X) := 1$.
- *Collapse*$(X_1, X_2, \ldots, X_c)$:



- *Quantile*(q): After collapse, returns $X[q \cdot k]$

Merging policies

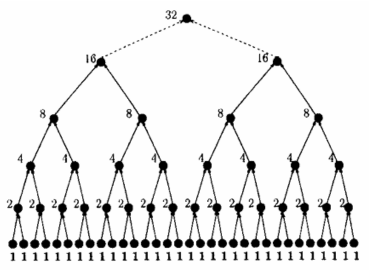


Figure: MP-sketch for $b = 6$ buffers.

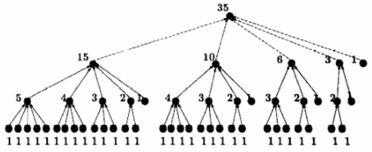


Figure: MRL-sketch for $b = 5$ buffers.

Our contribution

- Improve performance using its own predictions. If we have a suantile sketching algorithm, we can use it, to approximate the CDF.
- The slowest part is to sort the buffers on the first level, so try to improve this. After a sketch was built from scratch, we can build a second one, using the first.
- In every insertion, ask the first sketch what is the rank of that element. Then try to insert it into the buffers corresponding position.
- Then use a sorting algorithm that performs well on nearly sorted data (like insertion sort).

Measurements

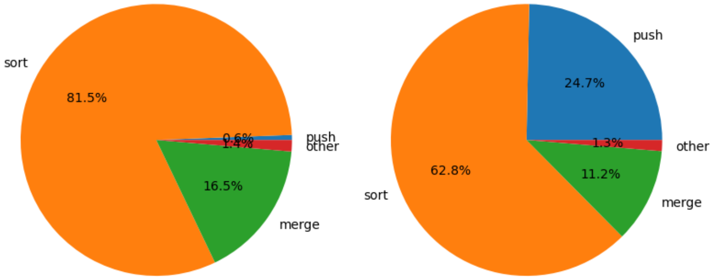
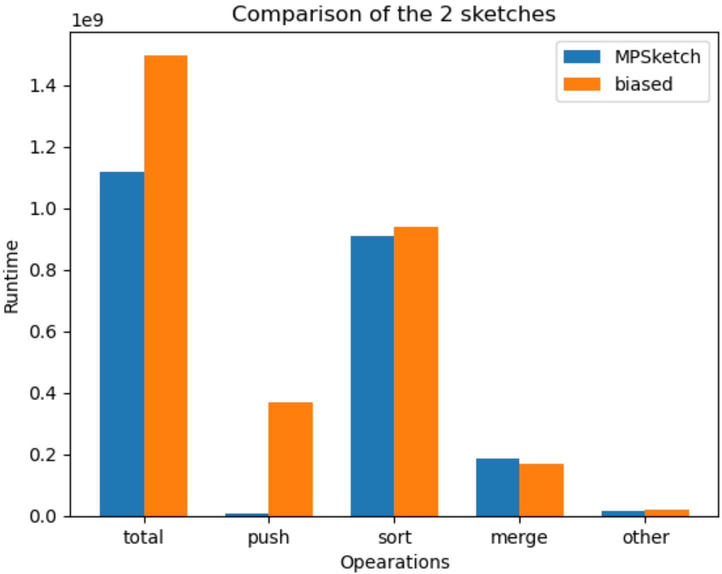


Figure: Operation proportions to the runtime of sketches. The original is on the right, the biased is on the left.
$n = 10^5, \epsilon = 0.001, b = 6, k = 3125$

Measures



Comparison of the 2 sketches
1e9
MPSketch
biased
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Runtime
total
push
sort
merge
other
Opearations

Future plans

- Instead of sorting the buffers, we can use a smarter data structure for insertion, such as a skip list.
- Examine relative error sketching algorithms such as DDSketch, and ReqSketch. Furthermore explore the literature on *some quantile* sketches.

Thank you for your attention!